Operating System Services

Narayan Changder//Abhisek Roy

February 29, 2016

Overview

1. Basic system calls
2. Operating environment
3. Processes
4. Timers
5. Signal handling
6. Error reporting
7. Users and passwords

1. Environment Variables
   1. os.environ - A dictionary containing current environment variables
   2. user = os.environ['USER']
   3. os.environ['PATH'] = "/bin:/usr/bin"
   4. Current directory and umask

   1. os.chdir(path)# Change current working directory
   2. os.getcwd()# Get current working directory
   3. os.umask(mask)# Change umask setting. Returns previous umask
2. User and group identification
   1. os.getegid()
   2. os.geteuid()
   3. os.getgid()
   4. os.getuid()
   5. os.setgid(gid)
   6. os.setuid(uid)

Example

Example

Listing 1: Process

```
import os
print("All Environmental variables", os.environ)
print("Details of only PATH variable", os.environ['PATH'])
print("Current working directory", os.getcwd())
print("Change of directory")
os.chdir("/home/abhisekroy/PycharmProjects/")
os.chdir('/home/abhisekroy/PycharmProjects/PythonOSprocess')
print(os.getcwd())
print("Group ID connected with present process", os.getgid())
print("User ID connected with present porcess", os.getuid())
print("List of group IDS", os.getgroups())
print("Get present process ID", os.getpid())
print("Get parent process ID", os.getppid())
print("execution of system call ps", os.system("ps"))
print("execution of system call ls -l", os.system("ls -l"))
```

Example

Example

Listing 2: Process creation

```
import os
import time
MAX_COUNT = 5
c_PID = os.fork()
pid = os.getpid()
for i in range(MAX_COUNT):
    print("This line is from pid %d, Parent PID:
    %d, value = %d\n" %(pid, os.getppid(), i))
    time.sleep(2)

Output:
This line is from pid 6247, Parent PID: 3714, value = 0
This line is from pid 6248, Parent PID: 6247, value = 0
This line is from pid 6247, Parent PID: 3714, value = 1
This line is from pid 6248, Parent PID: 6247, value = 1
```

Example

Example

Listing 3: Process creation

```
import os
import time
List = [10, 20, 30, 40, 50, 60]
sum = 0
sub = 0
PID = os.fork()
if PID == 0:
    for i in range(len(List)):
        sum = sum + List[i]
    print("Value of addition = %d by process with ID:
    %d" %(sum, os.getpid()))

elif PID > 0:
    os.wait()
    for i in range(len(List)):
        sub = sub - List[i]
```

Example

Example

Listing 4: Process creation

```
            print("Value of subtraction = %d by process with ID:
        %d" %(sub, os.getpid()))
else:
    print("Process creation is unsuccessful")
print("Thank you")
Output:
Value of addition = 210 by process with ID:6733
Thank you
Value of subtraction = -210 by process with ID:6732
Thank you
```

Example

Example

Listing 5: Process creation

```
from multiprocessing import Process
import os
import time
def sleeper(name, seconds):
    print('starting child process with id: ', os.getpid())
    print('parent process:', os.getppid())
    print('sleeping for %s ' % seconds)
    time.sleep(seconds)
    print("Done sleeping")
```

Example

Example

Listing 6: Process creation

```
print("in parent process (id %s)" % os.getpid())
p = Process(target=sleeper, args=('bob', 5))
p.start()
print("in parent process after child process start")
print("parent process about to join child process")
p.join()
print("in parent process after child process join")
print("parent process exiting with id ", os.getpid())
print("The parent's parent process:", os.getppid())
Output:
in parent process (id 6923)
in parent process after child process start
parent process about to join child process
starting child process with id:  6924
parent process: 6923
sleeping for 5
Done sleeping
in parent process after child process join
```

Inter Process Communication by Queue

Example

Listing 7: IPC

```
from multiprocessing import Process, Queue
import os
import time
def f(q):
    time.sleep(5)
    q.put([42, None, 'hello'])
    print("Child is completed")

q = Queue()
p = Process(target=f, args=(q,))
p.start()           # Start the execution of child process p.
print("child process with PID = %d" %(p.pid))
os.waitpid(p.pid, 0) # wait the parent process for process p.
Otherwise parent process continues.
print("parent process with PID = %d PPID = %d" %(os.getpid(),
```

Example

Example

Listing 8: Process creation

```
os.getppid()))
print(q.get())
print("Parent Complete")
Output:
child process with PID = 9123
Child is completed
parent process with PID = 9122 PPID = 3714
[42, None, 'hello']
Parent Complete
```

Zombie

Example

Listing 9: Process Zombie

```
from multiprocessing import Process

def say_hello(name='world'):
    print "Hello, %s" % name

p = Process(target=say_hello)
p.start()
p.join()
```

We import the Process class, create a function the process will run, then instantiate a Process object with the function it should run. Nothing has happened yet and won't until we tell it to begin via p.start(). The process will run and return it's result. Finally, we tell the process to complete via p.join(). NOTE - Without the p.join(), the child process will sit idle and not terminate, becoming a zombie

Orphan process

Example

Listing 10: Process orphan

```
import os
def child():
    print("in child:", os.getpid())
    print("Its parent ID:", os.getppid())

def parent():
    newpid = os.fork()
    if newpid == 0:
        child()
    print("in parent::", os.getpid())
    os._exit(0)

parent()

Output:
('in parent::', 5770)
('in child:', 5771)
('Its parent ID:', 1)
```

Inter Process Communication by Pipe

IPC

Multiprocessing module has two communication channels :

1. About Pipe():
   - Returns a pair of connection objects connect by a pipe.
   - Every object has send/recv methods that are used in the communication between processes.
   - Note that data in a pipe may become corrupted if two processes (or threads) try to read from or write to the same end of the pipe at the same time.
   - Of course there is no risk of corruption from processes using different ends of the pipe at the same time.
2. About Queue():
   - Returns a process shared queue.
   - Any pickle-able object can pass through it.
   - Thread and process safe.
   - put() for writing an element in the queue and get() is used to retrieve an element from queue.

Sharing state between processes

Shared memory :

Sharing state between processes:

1. Shared memory :
   - Python provide two ways for the data to be stored in a shared memory map.
   - Data can be stored in a shared memory map using Value() or Array().
   - 'd' indicates a double precision float and 'i' indicates a signed integer.
   - Shared objects will be process and thread-safe.
2. Server process:
   - A Manager object control a server process that holds python objects and allow other process to manipulate them.
   - What is Manager ?
     - Controls server process which manages shared object.
     - It make sure the shared object get updated in all processes when anyone modifies it.

Inter Process Communication by Pipe

Example

Listing 11: IPC-Pipe

```
from multiprocessing import Process, Pipe
import os
import time
def f(conn):
    time.sleep(5)
    conn.send([42, None, 'hello']) # child writes the list on pipe by
    conn.close() # closing of child descriptor
    print("Child is completed")
parent_conn, child_conn = Pipe() #creates two descriptor for
parent amd child process respectively for comm, through pipe.
p = Process(target=f, args=(child_conn,)) # child performs
writes on pipe.
p.start()           # Start the execution of child process p.
print("child process with PID = %d" %(p.pid))
```

Inter Process Communication by Pipe

Example

Listing 12: Process creation

```
os.waitpid(p.pid, 0) # wait the parent process for child process p.
 Otherwise parent process continues.
print("parent process with PID = %d PPID = %d" %(os.getpid(),
os.getppid()))
print(parent_conn.recv()) # parent reads the list from pipe by
 parent_conn through recv()
print("Parent Complete")
parent_conn.close() # closing of parent descriptor


Output:
child process with PID = 9953
Child is completed
parent process with PID = 9952 PPID = 3714
[42, None, 'hello']
Parent Complete
```

Program with Shared Memory

Example

Listing 13: Process creation

```
from multiprocessing import Value, Array
import os

num = Value('d', 3.54678)
arr = Array('i', range(10))
print(num.value)
print (arr[:])
C_PID = os.fork()
if C_PID == 0:
        child_PID = os.getpid()
        print("Child process with PID=%d with parent ID=%d"
        %(child_PID, os.getppid()))
        num.value = 5.567
        for i in range(len(arr)):
            arr[i] = arr[i] + 1
        print(num.value)
        print (arr[:])
```

Program with Shared Memory

Example

Listing 14: Process creation

```
elif C_PID > 0:
        os.wait()        # Waiting for all child processes
        parent_ID = os.getpid()
        print("Parent process with PID=%d with parent ID=%d"
        %(parent_ID, os.getppid()))
        num.value = 10.234
        for i in range(len(arr)):
            arr[i] = arr[i] + 5
        print(num.value)
        print (arr[:])
else:
        print("New process is not created")
```

Program with Shared Memory

Example

Listing 15: Process creation

```
Output:
3.54678
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Child process with PID=11024 with parent ID=11023
5.567
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Parent process with PID=11023 with parent ID=3714
10.234
[6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
```

program by server process

Example

Listing 16: Process creation

```
from multiprocessing import Process, Manager
def f(d, l):
    d[1] = '1'
    d['2'] = 2
    d[0.25] = None
    l.reverse()
    print(d)
    print(l)
if __name__ == '__main__':
    manager = Manager()
    d = manager.dict()
    l = manager.list(range(10))
    p = Process(target=f, args=(d, l))
```

program by server process

Example

Listing 17: Process creation

```
    p.start()
    p.join()
    d[1] = '5'
    d['2'] = 4
    d[0.25] = '1'
    print(d)
    print(l)

Output:
{0.25: '1', 1: '5', '2': 4}
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
{0.25: None, 1: '1', '2': 2}
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```